Journal of Machine Learning Research 1 (2014) 1-48 Submitted 6/14; Published 12/14

Refinery: An Open Source Topic Modeling Web Platform

Daeil Kim DAEIL.KIM@NYTIMES.COM
Benjamin F. Swanson DUJIAOZHU@GMAIL.COM
Michael C. Hughes MHUGHES@CS.BROWN.EDU
Erik B. Sudderth SUDDERTH@CS.BROWN.EDU

Editor:

Abstract

We develop an open source web platform for the application of topic models to large document collections called Refinery. The package is built in Python and acts as a standalone web application that is driven entirely by its graphical interface. The machine learning is driven by BNPy, a Bayesian nonparametric toolbox written also in Python combined with a set of interactive visualizations built using d3.js. Furthermore, Refinery contains a phrase extraction step. This allows for users to query documents by collecting phrases that might of relevance to them. More details can be found at `http://daeilkim.com/refinery.html`

Keywords: Topic modeling, Python, Visualization, BNPy, Flask, d3

1. Introduction

Topic models have become a ubiquitous class of tools for the analysis of large document collections. However, there is still a significant barrier when it comes to their application for individuals who have very little to no background in coding or computer science. For example, journalists could potentially benefit from the rich semantic themes that topic models are capable of producing for exploring larget datasets generated by FOIA (Freedom of Information Act) requests, but very few have the expertise to implement the code necessary to execute these models, let alone an intuition for how to fine tune its parameters. Even if the application is successful, there is an added difficulty in making sense of the large number of topics that are generated as well as accessing relevant documents that might be of interest for a learned topic.

To make these types of models and their results more accessible, we built an open source web application called Refinery that is meant to be deployed locally using Vagrant and VirtualBox. Installation is simple conditioned on having a few packages installed. Our main imagined audience when building this tool was driven primarily by journalists, who represent a specific, but valuable audience segment that helped constrain and guide our design choices. Since most journalists do not have a computer science background, our application must cater to an interface that they can understand without inundating them with functionality that might act to simply confuse or obstruct their use cases.

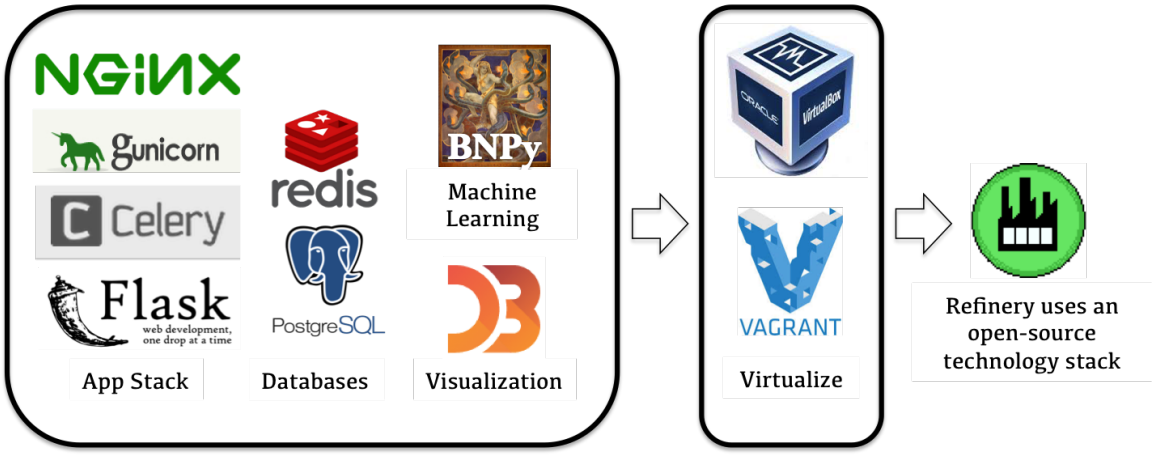

Figure 1: Refinery is built off of several software modules. After installation, the app can be accessed from its default address of `http://11.11.11.11:8080` using the default username of "doc" and password "refinery". For a full video demo of the app, please take a look at `http://youtu.be/7yRQ1J9Z_LI`.

2. Running Refinery

Refinery is a full stack web application built mainly off of Python. It uses several technologies as shown in Fig.1, but was developed with the requirement that its implementation process be as simple as possible. Refinery requires three main packages - Git (`http://git-scm.com`), Virtualbox (`http://www.virtualbox.org`) and Vagrant VM (`http://www.vagrantup.com`). Git is necessary since we'll use this to clone the repository that will contain the main source code. VirtualBox and Vagrant VM allows Refinery to exist within a virtual machine that is accessible through your browser. The Vagrant package allows for the deployment of a Puppet manifest (`http://www.puppetlabs.com`), which allows for the automation of a large number of software modules that are necessary for this application. To modify the installation process, the configuration file **Vagrantfile** located within the root directory contains parameters that can allow for the modification of puppet manifests as well as the default web address used to access the application. Changes to the Puppet manifests can allow for other software modules to be installed, but we warn against such changes unless the user is capable of such technical modifications. The process for the installation of Refinery is as follows from the command line:

```
> git clone https://github.com/daeilkim/refinery.git
> vagrant up
```

After logging into Refinery, a home screen appears that allows users to upload files. Currently, we require that documents come in the form of a zip file that when uncompressed contains a folder and within that folder a text file for each individual document.

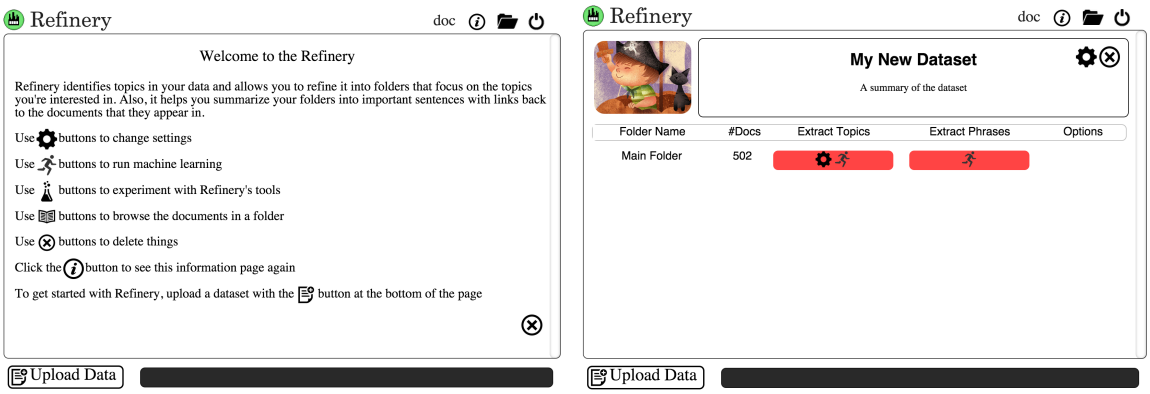

Figure 2: The home screen after logging in is displayed to the left with some initial instructions as to what each button does. One a dataset is uploaded (for these figures we pulled approximately 500 New York Times articles that contained the keyword "obama" during the year 2013) the resulting screen to the right appears. All the preprocessing that is typically done from the raw text is performed automatically in Refinery right after the upload process.

3. Topic Modeling

The topic modeling algorithm used for Refinery is drawn from BNPy (Bayesian Nonparametric Python Modeling Toolbox - `https://bitbucket.org/michaelchughes/bnpy-dev/`), which is a toolbox for the general implementation of Bayesian nonparametric graphical models Hughes and Sudderth (2014). In its current status, it is using Latent Dirichlet Allocation, Blei et al. (2003) with variational inference as the learning procedure though BNPy supports a number of advanced topic modeling algorithms which can help further extend the functionality of Refinery.

After the initial upload process, there will be two red buttons. The gear icon allows you to customize some parameters for the topic modeling algorithm and the running man icon begins the topic modeling process. Once the algorithm has finished running, the red button now changes to a green one and selecting the flask icon will shift the screen to the topic modeling view. Refinery displays the learned global topics as a multi-colored ring. The colored ring segments each represent the global topic proportion throughout the document corpus. Hovering over each segment displays the top 50 words associated with that topic.

Refinery also allows us to create document subsets that are related to a topic or several topics. Clicking on a colored segment inserts that particular topic in the black document selection box found in Fig. 3 and Refinery allows up to eight topics to be inserted as search queries. Additionally, kewyords can be inserted as well if we wish to specify more specific queries. The green circular scroll bar allows us to determine how much of the document subset that we wish to find for those given topics. Finally, when we are ready to create this document subset, we can select the *Create Document Subset* button below. This results in a list of documents with their topic proportions shown for that subset. Clicking on the document icon allows one to view the raw text as well.

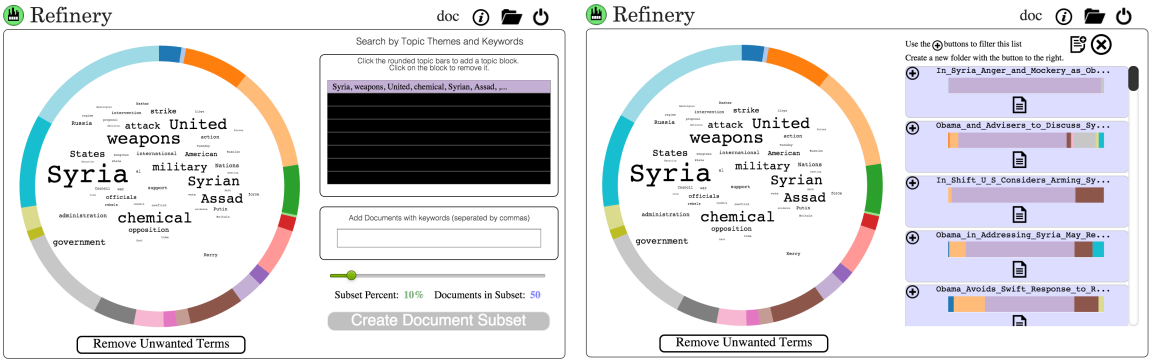

Figure 3: The results of a topic modeling analysis on approximately 500 New York Times articles that contained the keyword "obama" during the year 2013. In the figure above, the "Syria" topic has been selected and a document subset containing 50 documents have been returned. This subset can be further analyzed again using the topic modeling algorithm if we wish to discover more specific topics.

4. Phrase Extraction and Refinement

Refinery also implements a sentence boundary detection algorithm which uses a support vector machine with a linear kernel that is trained off of the Wall Street Journal and Brown corpuses Gillick (2009). The library is called **Splitta** and can be found at `https://code.google.com/p/splitta/`. The phrase extraction process begins by determining sentence boundaries using the library mentioned above and generating an initial diverse list of phrases. The user is then asked to choose phrases of interest that might be of relevance. This intial collection of phrases acts as the foundation for other future queries. From here, a user can then ask for either similar or dissimilar phrases or delete phrases that are no longer of relevance. By clicking on a given phrase, the full document is loaded as well as the phrase of interest being highlighted within the text.

5. Conclusion

Refinery was motivated by the need for non-technical researchers and professionals who were interested in applying the latest topic modeling algorithms to their document corpus. Several others have also created visualizations for topic models. Notably, Chaney and Blei (2012) created a web-based navigator to help users understand the relationship between topics and documents, while Chuang et al. (2012) developed a visualization package for assessing the goodness of topics. Refinery differs in several ways from these packages by simplifying and integrating the analysis of new document corpuses within the graphical user interface as well as focusing on design elements that simplify the search for relevant topics within these documents. Furthermore, we incorporate a phrase extraction step that allows for a more refined search across documents by using entire phrases as query terms.

Future updates will upgrade the current topic modeling algorithm from Latent Dirichlet Allocation to a memoized Hierarchical Dirichlet Process approach, Hughes et al. (2015) where the number of clusters do not have to be defined ahead of time. Furthemore, we

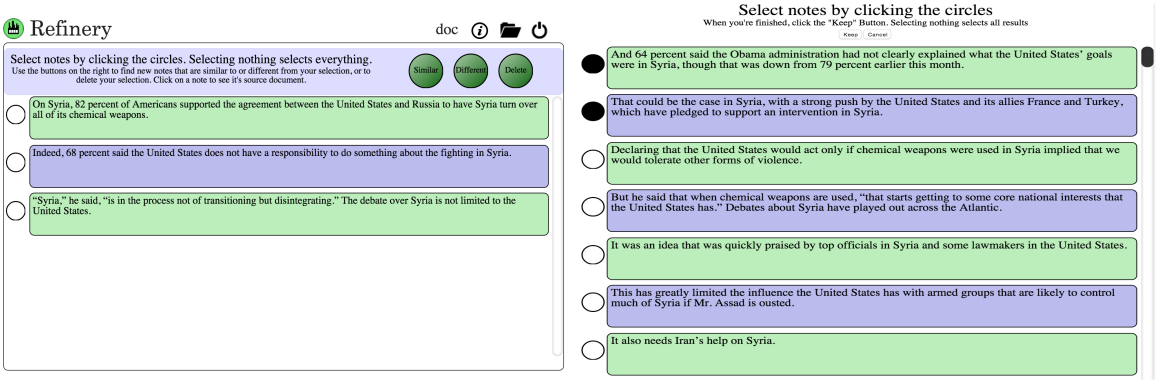

Figure 4: On the left, we've chosen an initial set of phrases related to news regarding the war in Syria. Hitting the green "similar" button on the upper right corner brings up the figure on the right which shows another set of phrases that might be similar to the collection of phrases we've already chosen.

hope to support a larger variety of potential documents for file types such as spreadsheets, powerpoint presentations, and word documents using the **Tika** library (`http://tika.apache.org/1.6/parser.html`).

Acknowledgments

Refinery was a recipient of the Knight Prototype Fund in 2014.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. ISSN 1532-4435. doi: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993. URL `http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993`.

Allison June-Barlow Chaney and David M Blei. Visualizing topic models. In *ICWSM*, 2012.

Jason Chuang, Christopher D. Manning, and Jeffrey Heer. Termite: Visualization techniques for assessing textual topic models. In *Advanced Visual Interfaces*, 2012. URL `http://vis.stanford.edu/papers/termite`.

Dan Gillick. Sentence boundary detection and the problem with the us. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 241–244. Association for Computational Linguistics, 2009.

Michael C. Hughes and Erik B. Sudderth. bnpy: Reliable and scalable variational inference for bayesian nonparametric models. In *NIPS - Probabilistic Programming Workshop*, 2014.

Michael C. Hughes, Daeil Kim, and Erik B. Sudderth. Reliable and scalable variational inference for the hierarchical dirichlet process. In *AISTATS*, 2015.